REMARKS

Claims 1-58 are pending in the application. In view of the following remarks, Applicant respectfully requests that this application be allowed and forwarded on to issuance.

The § 102 Rejections

Claims 1-14, 31-45 and 51-53 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 5,421,016 to Conner et al (hereinafter referred to as "Conner").

The § 103 Rejections

Claims 15-30, 46-50 and 54-58 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Conner in view of U.S. Patent No. 5,307,456 to MacKay.

Claims 1-15

Claim 1 recites a computer-implemented architecture comprising [emphasis added]:

- one or more first objects that support only static properties; and
- one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects*** to effect one or more property value changes on the one or more first objects in a manner that makes the one or more first objects appear as if they support dynamic properties.

In making out the rejection of this claim, the Office argues that this claim is directed to a "computer-implemented architecture for performing the method of

claim 51, and is similarly rejected under the same rationale." Applicant respectfully submits that claim 1 is not simply a "computer-implemented architecture for performing the method of claim 51." As such, the Office has failed to independently examine this claim.

However, in the interest of furthering prosecution of this case, Applicant will attempt to respond. In its rejection of claim 51, the Office equates Applicant's "one or more objects" with Conner's "application that was designed for statically defined classes." In the context of claim 1, Applicant believes that the Office intends to equate Conner's application with Applicant's "one or more ***first*** objects." Applicant is unclear what aspect of Conner the Office would equate with Applicant's "one or more second objects" because claim 51 contains no such language. Regardless, claim 1 recites "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***." Applicant has studied Conner and can find nothing that would anticipate or render obvious this feature. Because Conner does not disclose "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***," it cannot disclose that the one or more first objects are called to "effect one or more property value changes on the one or more first objects in a manner that makes the one or more first objects appear as if they support dynamic properties." Accordingly, for at least this reason, this claim is allowable.

Claims 2-15 depend from claim 1 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 1, are neither disclosed nor suggested in the references of record, either singly or in combination with one

another. Given the allowability of these claims, the rejection of claim 15 over the combination with MacKay is not seen to add anything of significance.

Claims 16-23

Claim 16 recites a multi-media project editing architecture comprising [emphasis added]:

- one or more first objects that support only static properties, the one or more first objects being configured to implement a transform associated with processing of a multi-media editing project;
- one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects*** to effect one or more property value changes on the one or more first objects in a manner that makes the one or more first objects appear as if they support dynamic properties; and
- one or more data structures associated with the one or more second objects, individual data structures containing property data that is to be used by the one or more second objects to effect a property value change.

In making out the rejection of this claim, the Office incorporates the rejection of claim 51. In its rejection of claim 51, the Office equates Applicant's "one or more objects" with Conner's "application that was designed for statically defined classes." In the context of claim 16, Applicant believes that the Office intends to equate Conner's application with Applicant's "one or more ***first*** objects." Applicant is unclear what aspect of Conner the Office would equate with Applicant's "one or more second objects" because claim 51 contains no such language. Regardless, claim 16 recites "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***." Applicant has studied Conner and can find nothing that would anticipate

or render obvious this feature. Because Conner does not disclose "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***," it cannot disclose that the one or more first objects are called to "effect one or more property value changes on the one or more first objects in a manner that makes the one or more first objects appear as if they support dynamic properties." In addition, MacKay does not supply the missing feature. Accordingly, for at least this reason, this claim is allowable.

Claims 17-23 depend from claim 16 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 16, are neither disclosed nor suggested by the references of record, either singly or in combination with one another.

Claims 24-30

Claim 24 recites a multi-media project editing architecture comprising [emphasis added]:

- a software-implemented matrix switch having multiple input pins and multiple output pins, the multiple input pins being routable to the multiple output pins, the switch being configured to provide a data stream that represents a multi-media project;
- a data structure associated with the matrix switch and configured for use in programming the matrix switch to provide a routing scheme for routing input pins to output pins;
- one or more first objects associated with the matrix switch, the one or more first objects supporting only static properties associated with rendering of a multi-media project;
- one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects*** to effect one or more property value changes on the one or more first objects

in a manner that makes the one or more first objects appear as if they support dynamic properties.

In making out the rejection of this claim, the Office incorporates the rejection of claim 51. In its rejection of claim 51, the Office equates Applicant's "one or more objects" with Conner's "application that was designed for statically defined classes." In the context of claim 24, Applicant believes that the Office intends to equate Conner's application with Applicant's "one or more ***first*** objects." Applicant is unclear what aspect of Conner the Office would equate with Applicant's "one or more second objects" because claim 51 contains no such language. Regardless, claim 24 recites "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***." Applicant has studied Conner and can find nothing that would anticipate or render obvious this feature. Because Conner does not disclose "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***," it cannot disclose that the one or more first objects are called to "effect one or more property value changes on the one or more first objects in a manner that makes the one or more first objects appear as if they support dynamic properties." In addition, MacKay does not supply the missing feature. Accordingly, for at least this reason, this claim is allowable.

Claims 25-30 depend from claim 24 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 24, are neither disclosed nor suggested by the references of record, either singly or in combination with one another.

Claims 31-40

Claim 31 recites a property value-changing method comprising [emphasis added]:

- providing one or more objects that support only static properties;
- providing one or more ***programmable objects*** configured to effect property value changes on the objects that support only static properties; and
- effecting at least one property value change on the one or more objects that support only static properties ***using the one or more programmable objects***.

In making out the rejection of this claim, the Office states that claim 31 "includes the same subject matter as in claim 51, and is similarly rejected under the same rationale." After linking the rejection of claim 31 to the rejection of claim 51, the Office then argues that Conner discloses effecting at least one property value change on the one or more objects that support only static properties using the one of more programmable objects. The Office cites to column 2, lines 22-43, and column 31, lines 5-17, which are reproduced below:

> The redispatch stub invokes the objects dispatch method passing it enough information to determine the correct method procedure in a dynamic manner. The dispatch function then invokes the appropriate method procedure and returns any return value to the redispatch stub which returns it to the original application. Thus, the original static method call can be converted to a dynamic call (a dispatch function call) without any change on the part of the calling application's binary image.
>
> Redispatch stubs can be used to allow a class with a definition that can vary at runtime to be used by an application that was designed for statically defined classes. This is done by replacing all the entries in the method procedure table for the dynamic class with the appropriate redispatch stub entries during initialization of the class object. Then, when the class includes suitable definitions for its dispatch functions, it can change the association of methods to

> method procedures at runtime without corrupting the execution of an application using the class. *Col. 2, lines 22-43.*
>
> Logic for providing a dynamic insertion of a replacement parent class, referred to in object programming as a parent class shadow, is detailed in this section of the invention. This processing allows the statically compiled definition of what parent class is linked to a particular class at runtime to be dynamically altered during execution. The ability to insert a new parent class into a statically compiled class hierarchy offers more flexibility to maintain and enhance existing code after it has appeared in binary form. It also offers a new degree of freedom for customizing code without access to source materials since this result can be achieved without recompilation. *Col. 31, lines 5-17.*

In its rejection of claim 51, the Office equates Applicant's "one or more objects that support only static properties" with Conner's "application that was designed for statically defined classes." Applicant is unclear what aspect of Conner the Office would equate with Applicant's "programmable object" because claim 51 contains no such language. Regardless, Applicant can find nothing in the cited excerpts, or anywhere else in Conner, that would disclose or suggest "providing one or more ***programmable objects*** configured to effect property value changes on the objects that support only static properties" as that term is used in the context of Applicant's specification. Because Conner does not disclose or suggest providing one or more programmable objects, as the Applicant uses the term, it cannot disclose or suggest "effecting at least one property value change on the one or more objects that support only static properties ***using the one of more programmable objects***." Accordingly, for at least this reason, this claim is allowable.

Claims 32-40 depend from claim 31 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited

features which, in combination with those recited in claim 31, are neither disclosed nor suggested by the references of record, either singly or in combination with one another.

Claims 41-44

Claim 41 recites a property value-changing method comprising [emphasis added]:

- programming a programmable object with property data that defines when certain property value changes are to be made and what those property value changes are;
- ***calling, with the programmable object, one or more objects that do not support dynamic properties***; and
- responsive to said calling, using the property data to effect a property value change on the one or more objects that do not support dynamic properties.

In making out the rejection of this claim, the Office states that claim 41 "includes the same subject matter as in claim 31, and is similarly rejected under the same rationale." The rejection of claim 31 states that "claim 31 includes the same subject matter as in claim 51, and is rejected under the same rationale." Applicant respectfully submits that each of these claims (claims 31, 41, and 51) merits its own independent examination, but Applicant will try to follow the Office's logic.

In its rejection of claim 51, the Office equates Applicant's "one or more objects that support only static properties" with Conner's "application that was designed for statically defined classes." In the context of claim 41, Applicant believes that the Office intends to equate Conner's application with Applicant's "one or more objects that do not support dynamic properties." Applicant is unclear

what aspect of Conner the Office would equate with Applicant's "programmable object" because claim 51 contains no such language. Regardless, claim 41 recites "***calling, with the programmable object, one or more objects that do not support dynamic properties.***" Applicant has studied Conner and can find nothing that would anticipate or render obvious this feature. Because Conner does not disclose "calling, with the programmable object, one or more objects that do not support dynamic properties," it cannot disclose, responsive to the act of calling, "using the property data to effect a property value change on the one or more objects that do not support dynamic properties." Accordingly, for at least this reason, this claim is allowable.

Claims 42-44 depend from claim 41 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 41, are neither disclosed nor suggested by the references of record, either singly or in combination with one another.

Claim 45

Claim 45 recites one or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to [emphasis added]:

- provide one or more objects that support only static properties;
- provide one or more programmable objects configured to effect property value changes on the objects that support only static properties;

program the one or more programmable objects with property data that is to be used by the one or more programmable objects to effect said at least one property value change, the property data

comprising: property value changes that are to be made, time(s) at which property value changes are to be made, and how the property value changes are to be made; and

effect at least one property value change on the one or more objects that support only static properties ***by using the one or more programmable objects to call the one or more objects that support only static properties.***

In making out the rejection of this claim, the Office states that claim 45 "includes the same subject matter as in claim 31, and is similarly rejected under the same rationale." The rejection of claim 31, in turn, states that "claim 31 includes the same subject matter as in claim 51, and is similarly rejected under the same rationale." After linking the rejection of claim 45 to the rejections of claims 31 and 51, the Office then argues that Conner discloses effecting at least one property value change on the one or more objects that support only static properties by using the one of more programmable objects to call the one or more objects that support only static properties. The Office cites to column 2, lines 32-43, and column 31, lines 5-17, which are reproduced below:

> Redispatch stubs can be used to allow a class with a definition that can vary at runtime to be used by an application that was designed for statically defined classes. This is done by replacing all the entries in the method procedure table for the dynamic class with the appropriate redispatch stub entries during initialization of the class object. Then, when the class includes suitable definitions for its dispatch functions, it can change the association of methods to method procedures at runtime without corrupting the execution of an application using the class. *Col. 2, lines 32-43.*

> Logic for providing a dynamic insertion of a replacement parent class, referred to in object programming as a parent class shadow, is detailed in this section of the invention. This processing allows the statically compiled definition of what parent class is linked to a particular class at runtime to be dynamically altered during execution. The ability to insert a new parent class into a statically compiled class hierarchy offers more flexibility to maintain and

> enhance existing code after it has appeared in binary form. It also offers a new degree of freedom for customizing code without access to source materials since this result can be achieved without recompilation. *Col. 31, lines 5-17.*

In its rejection of claim 51, the Office equates Applicant's "one or more objects that support only static properties" with Conner's "application that was designed for statically defined classes." Applicant is unclear what aspect of Conner the Office would equate with Applicant's "programmable object" because claim 51 contains no such language. Regardless, Applicant can find nothing in the cited excerpts, or anywhere else in Conner, that would disclose or suggest effecting "at least one property value change on the one or more objects that support only static properties ***by using the one of more programmable objects to call the one or more objects that support only static properties."*** Accordingly, for at least this reason, this claim is allowable.

Claims 46-50

Claim 46 recites a property value-changing method comprising [emphasis added]:

- programming a programmable object with property data that defines when certain property value changes are to be made and what those property value changes are, the property value changes being associated with rendering of a multi-media editing project;
- ***calling, with the programmable object, one or more objects that do not support dynamic properties***; and
- responsive to said calling, using the property data to effect a property value change on the one or more objects.

In making out the rejection of this claim, the Office states that claim 46 "includes the same subject matter as in claim 31, and is similarly rejected under the same rationale." The rejection of claim 31 states that "claim 31 includes the same subject matter as in claim 51, and is rejected under the same rationale." Applicant respectfully submits that each of these claims (claims 31, 46, and 51) merits its own independent examination, but Applicant will try to follow the Office's logic.

In its rejection of claim 51, the Office equates Applicant's "one or more objects that support only static properties" with Conner's "application that was designed for statically defined classes." Applicant is unclear what aspect of Conner the Office would equate with Applicant's "programmable object" because claim 51 contains no such language. Regardless, claim 46 recites "***calling, with the programmable object, one or more objects that do not support dynamic properties***." Applicant has studied Conner and can find nothing that would anticipate or render obvious this feature. Because Conner does not disclose "calling, with the programmable object, one or more objects that do not support dynamic properties," it cannot disclose, responsive to the act of calling, "using the property data to effect a property value change on the one or more objects." In addition, MacKay does not supply the missing feature. Accordingly, for at least this reason, this claim is allowable.

Claims 47-50 depend from claim 46 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 46, are neither disclosed nor suggested by the references of record, either singly or in combination with one another.

Claims 51-54

Claim 51 recites a property value-changing method comprising [emphasis added]:

- providing one or more objects that support only static properties; and
- simulating dynamic properties with the one or more objects by changing one or more property values ***at a pre-programmed time***.

In making out the rejection of this claim, the Office argues that Conner discloses simulating dynamic properties with the one or more objects by changing one or more property values at a pre-programmed time. In particular, the Office states that "the original static method call can be converted to a dynamic call . . . this is done by replacing all the entries in the method procedure table for the dynamic class with the appropriate redispatch stub entries during initialization of the class object." The Office then cites to column 2, lines 12-43, column 31, lines 5-17, and column 33, lines 6-22, all of which are reproduced below [emphasis added]:

> These and other objectives of the present invention are accomplished by the operation of an algorithm in the memory of a processor that uses redispatch method stubs to convert static method calls into dynamic method calls. When the Systeom [sic] Object Model (SOM) compiler generates support for defining a class, it generates a redispatch stub for each method defined in the class. A redispatch stub is a short sequence of instructions that has the exact same calling sequence as its corresponding method.
>
> The redispatch stub invokes the objects dispatch method passing it enough information to determine the correct method procedure in a dynamic manner. The dispatch function then invokes the appropriate method procedure and returns any return value to the redispatch stub which returns it to the original application. Thus, the original static

method call can be converted to a dynamic call (a dispatch function call) without any change on the part of the calling application's binary image.

Redispatch stubs can be used to allow a class with a definition that can vary at runtime to be used by an application that was designed for statically defined classes. This is done by replacing all the entries in the method procedure table for the dynamic class with the appropriate redispatch stub entries ***during initialization of the class object***. Then, when the class includes suitable definitions for its dispatch functions, it can change the association of methods to method procedures at runtime without corrupting the execution of an application using the class. *Col. 2, lines 12-43.*

Logic for providing a dynamic insertion of a replacement parent class, referred to in object programming as a parent class shadow, is detailed in this section of the invention. This processing allows the statically compiled definition of what parent class is linked to a particular class at runtime to be dynamically altered during execution. The ability to insert a new parent class into a statically compiled class hierarchy offers more flexibility to maintain and enhance existing code after it has appeared in binary form. It also offers a new degree of freedom for customizing code without access to source materials since this result can be achieved without recompilation. *Col. 31, lines 5-17.*

The invention consists of a programming mechanism called redispatch stubs to ameliorate the difference between static and dynamic models. A redispatch stub is a small procedure with an entry point that can be placed into a table of procedure entry points. The table of procedure entry points are used in a static object model as a substitute for the actual method entry point that is expected. The redispatch stub is generated automatically based on the requirements of the dynamic object model. The redispatch stub converts the call generated in the static object model into the form necessary in the dynamic object model and supplies any missing information in the process. Thus, if an object is accessed from a static object model that is provided by a dynamic object model, it can be represented to the static object model via a table of entry points which each indicate a particular redispatch stub. *Col. 33, lines 6-22.*

As discussed in the excerpts cited, Conner replaces all the entries in the method procedure table for the dynamic class with the appropriate redispatch stub entries ***during initialization of the class object***. In contrast, Applicant simulates dynamic properties with one or more objects that support only static properties by changing one or more property values ***at a pre-programmed time***. An example of how this might work is given in the specification at page 54, lines 1-24, reproduced below:

> Consider the following example where object 4000 comprises a ball. Consider that, for the ball, we want the helper object to cause the ball to be red at time 0, change immediately to blue at time 2, and blend into yellow at time 4. An exemplary data structure that can enable this to be done is shown in Fig. 40. In this example, the data structure "(0, Red, Jump)" indicates that at time 0, the ball is to immediately (i.e. "jump to") be red; the structure "(2, Blue, Jump)" indicates that at time 2, the ball is to immediately be blue; and the structure "(4, Yellow, Interpolate)" indicates that at time 4, the ball is to change in a linear fashion (i.e. interpolate) into yellow from blue.
>
> The helper object 4004 is programmed (or pre-programmed) with the various data structures that define the desired properties, their value changes, and the time and manner in which the value change is to take place. The helper object keeps track of the time, by for example, being called or otherwise notified of the time. At the appropriate times, the helper object, using the array of structures, calls the object whose property values are desired to be changed so that the property values can be changed. In the present example, at time 0, the helper object calls object 4000 or a property object on the object itself, and causes the property value for the color property to be set to "red". At time 2, the helper object 4004 calls the object 4000 and causes the property value for the color property to be set to "blue". At time 4, the helper object calls the object 4000 and causes the property value for the color property to interpolate from its previous value (i.e. blue) to the present value, i.e. yellow.
>
> Thus, object 4000 appears as if it supports dynamic properties when, in fact, it only supports static properties.

Thus, Conner neither discloses nor suggests simulating dynamic properties with one or more objects that support only static properties by changing one or more property values ***at a pre-programmed time***. Accordingly, for at least this reason, this claim is allowable.

Claims 52-54 depend from claim 51 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 51, are neither disclosed nor suggested by the references of record, either singly or in combination with one another. Given the allowability of these claims, the rejection of claim 54 over the combination with MacKay is not seen to add anything of significance.

Claims 55-58

Claim 55 recites a multi-media system comprising [emphasis added]:

- an application program configured to enable a user to define a multi-media project in which multiple digital source streams can be combined;
- a software-implemented matrix switch having multiple input pins and multiple output pins, the input pins being individually associated with inputs that can compete, during a common time period, for a particular output pin that is associated with the matrix switch, the switch being configured to receive, at its input pins, digital source streams;
- a first data structure associated with the matrix switch and configured for use in programming the matrix switch to provide a routing scheme for routing input pins to output pins such that at any given time, only one input pin is routed to the particular output pin;
- a second data structure associated with and different from the first data structure, the second data structure representing a user-defined multi-media project and being configured so that the first data structure can be derived therefrom;

one or more first objects associated with the matrix switch, the one or more first objects supporting only static properties associated with rendering of a multi-media project; and

- one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects*** to effect one or more property value changes on the one or more first objects in a manner that makes the one or more first objects appear as if they support dynamic properties.

In making out the rejection of this claim, the Office incorporates the rejection of claim 51. In its rejection of claim 51, the Office equates Applicant's "one or more objects" with Conner's "application that was designed for statically defined classes." In the context of claim 55, Applicant believes that the Office intends to equate Conner's application with Applicant's "one or more ***first*** objects." Applicant is unclear what aspect of Conner the Office would equate with Applicant's "one or more second objects" because claim 51 contains no such language. Regardless, claim 55 recites "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***." Applicant has studied Conner and can find nothing that would anticipate or render obvious this feature. Because Conner does not disclose "one or more second objects associated with the one or more first objects and configured to ***call the one or more first objects***," it cannot disclose that the one or more first objects are called to "effect one or more property value changes on the one or more first objects in a manner that makes the one or more first objects appear as if they support dynamic properties." In addition, MacKay does not supply the missing feature. Accordingly, for at least this reason, this claim is allowable.

Claims 56-58 depend from claim 55 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited

features which, in combination with those recited in claim 55, are neither disclosed nor suggested by the references of record, either singly or in combination with one another.

Conclusion

Applicant submits that all of the claims are in condition for allowance and respectfully requests that the Office pass the application along to issuance. If the Office's next anticipated action is to be anything other than issuance of a Notice of Allowability, Applicant respectfully requests a telephone call for the purpose of scheduling an interview.

Respectfully Submitted,

Dated: 4/8/04

By: _____

Lance R. Sadler
Reg. No. 38,605
(509) 324-9256